GPSAF: A Generalized Probabilistic Surrogate-Assisted Framework for Constrained Single- and Multi-objective Optimization

Julian Blank and Kalyanmoy Deb

COIN Report Number 2022004

***Abstract*—Significant effort has been made to solve computationally expensive optimization problems in the past two decades, and various optimization methods incorporating surrogates into optimization have been proposed. Most research focuses on either exploiting the surrogate by defining a utility optimization problem or customizing an existing optimization method to use one or multiple approximation models. However, only a little attention has been paid to generic concepts applicable to different types of algorithms and optimization problems simultaneously. Thus this paper proposes a generalized probabilistic surrogate-assisted framework (GPSAF), applicable to a broad category of unconstrained and constrained, single- and multi-objective optimization algorithms. The idea is based on a surrogate assisting an existing optimization method. The assistance is based on two distinct phases, one facilitating exploration and another exploiting the surrogates. The exploration and exploitation of surrogates are automatically balanced by performing a probabilistic knockout tournament among different clusters of solutions. A study of multiple well-known population-based optimization algorithms is conducted with and without the proposed surrogate assistance on single- and multi-objective optimization problems with a maximum solution evaluation budget of 300 or less. The results indicate the effectiveness of applying GPSAF to an optimization algorithm and the competitiveness with other surrogate-assisted algorithms.**

***Index Terms*—Surrogate-Assisted Optimization, Model-based Optimization, Simulation Optimization, Evolutionary Computing, Genetic Algorithms.**

I. Introduction

MANY optimization problems are computationally expensive and require the execution of one or multiple time-consuming objective and constraint functions to evaluate a solution. Expensive optimization problems (EOPs) are especially important in practice and are omnipresent in all kinds of research and application areas, for instance, Agriculture [1], Engineering [2], Health Care [3], or Computer Science [4]. Often the expensive solution evaluations (ESEs) are caused by running a simulation, such as Computational Fluid Dynamic [5], Finite Element Analysis [6], or processing a large amount of data [7], [8]. Most of such these simulation-based or data-intensive ESEs are black-box in nature [9] and no gradient information is available or even more time-consuming to derive. Thus, the optimization method must be designed for a significantly limited evaluation budget without any assumptions about the problem's fitness landscape.

Julian Blank and Kalyanmoy Deb are at Michigan State University, East Lansing, USA (email: {blankjul,kdeb}@msu.edu).

The majority of methods proposed for solving EOPs incorporate so-called surrogate models (or metamodels) [10] into the optimization procedure. The surrogate model provides an approximate solution evaluation (ASE) with less computational expense to improve the convergence behavior. A well-known research direction with the significant effort being made in the past is referred to as "Efficient Global Optimization" (EGO) [11]. In EGO, solutions are evaluated in each iteration based on the optimum of a utility optimization problem – also known as infill criterion – commonly defined by the surrogate's value and error predictions from Kriging [12]. The method can be summarized as a *fit-define-optimize* procedure: A surrogate model is fitted, a utility problem based on the surrogate predictions is defined, and then optimized to obtain an infill solution. Original limitations such as the evaluation of a single point per iteration, the lack of constraint handling, or dealing with multiple objectives have been investigated, and extensions have been proposed [13], [14]. Nevertheless, adding such additional requirements further complicates the utility problem and makes it significantly more challenging. The surrogate's role in such *fit-define-optimize* methods is *critical* because of the direct dependency on the infill criteria defined based on approximations and error predictions.

Another research direction pursued by researchers is the organic incorporation of surrogates into an existing optimization [15]. Such approaches aim to improve the convergence behavior of a "baseline" algorithm and, thus, the anytime performance. Researchers have explored different ways of incorporating surrogates into well-known population-based algorithms, such as genetic algorithms (GA) [16], differential evolution (DE) [17], or particle swarm optimization (PSO) [18]. All surrogate-assisted algorithms must find a reasonable trade-off between exploiting the knowledge of the surrogate and still exploring the search space. On the one hand, researchers have investigated methods adding a surrogate with lighter influence on the original algorithm, for instance, using surrogate-based pre-selection in evolutionary strategy [19] or a predictor for the individual replacement in DE [20]. On the other hand, the behavior of an existing algorithm might almost entirely rely on the surrogate predictions and guide the search significantly. For instance, a global and local surrogate have been incorporated into PSO to solve expensive large-scale optimization problems [21] or into DE for expensive constrained optimization problems [22]. The existence of numerous variants of surrogate-assisted algorithms indicates that

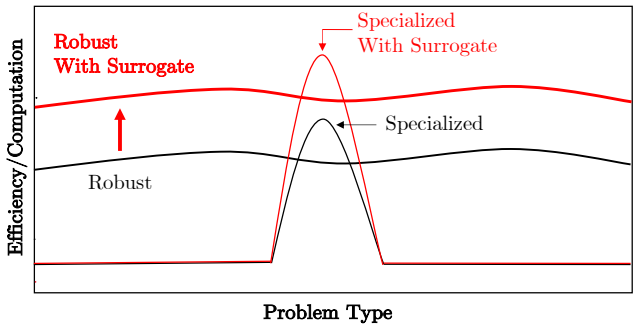

Fig. 1. Robustly Adding Surrogate-Assistance to Population-Based Algorithms (Illustration inspired from [16]).

many different ways of using surrogates during optimization method exist, but also that no best practice procedure has been established yet [23].

The need for more generalizable concepts in surrogate-assisted optimization has been identified, and frameworks aiming to solve a broader category of optimization problems have been proposed [24], [25]. Existing frameworks provide a generic method for different problems types using the *fit-define-optimize* strategy or by replacing locally optimized solutions (on ASEs) before their computationally expensive evaluation. One shortcoming of existing frameworks for solving EOPs is their design being primarily algorithm-dependent and their limitations to transferring it to different optimization methods. Thus, this matter shall be addressed in this paper by proposing a novel generalized probabilistic surrogate-assisted framework (GPSAF), adding surrogate assistance to population-based algorithms. In contrast to other surrogate-assisted algorithms that customize a *specific* algorithm, our goal is to provide a scheme to add surrogate assistance to a whole algorithm class. Figure 1 – inspired from [16] – shows different problem types on the x and the efficiency of algorithms on the y-axis. Whereas specific algorithms can be customized and thus be a specialist for a specific problem type, this study considers algorithms that can solve a broad category of problem types and adds surrogate assistance to them. Even though specialized surrogate-assisted algorithms are likely to outperform a generic concept on a specific problem type, the merit of this study is its broad applicability. The main contributions of this paper can be summarized as follows:

(i) We provide a categorization of existing surrogate-assisted algorithms regarding their surrogate usage. Methods are distinguished based on their surrogate's impact and importance during optimization and identify what has been less attention paid to in the past.

(ii) We propose a framework that applies to all kinds of population-based optimization algorithms. The framework enables existing optimization methods designed for unconstrained or constrained, single or multi-objective optimization to become assisted by a surrogate. Intuitive hyper-parameters can control the surrogate's impact. A specific setting to even disable the surrogate usage entirely exists, which demonstrates the truly surrogate assistant behavior.

(iii) We propose a novel way of dealing with surrogate prediction error algorithmically. In contrast to existing surrogate-assisted methods, we are using the search pattern of the search of an algorithm on the surrogate instead of only using final solutions. The prediction error of the surrogates and the search space exploration is addressed using a probabilistic knockout tournament selection. The surrogate prediction error is incorporated into the tournament selection and reliably balances the exploitation-exploration trade-off based on the surrogate's accuracy.

In the remainder of this paper, we first discuss related work of surrogate-assisted optimization and its challenges in Section II before proposing the generalized probabilistic surrogate-assisted framework (GPSAF) in Section III. A comparison of GPSAF and other state-of-the-art surrogate-based methods is provided in Section IV. Finally, conclusions are drawn, and future work is presented in Section V.

II. Background

In this section, the brief overview of methods in the previous section shall be enriched with details, and surrogate-assisted algorithms are categorized regarding their surrogate incorporation.

Surrogate-assisted methods can be roughly put into one of the following categories based on the surrogate's involvement: aided, customized, centered, or once (see Figure 2). The latter describes the early development of optimization using an approximation model, fitted exactly once during optimization and never updated (once). Algorithms that perform an update of the surrogate can mostly depend on its predictions (centered) or use it as an assistant in an existing method to improve the convergence behavior (aided, customized). The surrogate's role and dependency on the algorithm's design are vital for generalization and, thus, shall be spent special attention to. Next, a thematic overview of these different types of surrogate involvements in an algorithm is given.

Especially in the early phase of surrogate-based optimization, the surrogate was fitted *only once* and optimized. Thus, the optimization's outcome entirely depends on the accuracy of the surrogate model, assuming an efficient optimizer is used to obtain the surrogate's optimum.

The limitation of fitting a surrogate only once has soon been overcome by a more adaptive approach known as EGO (Efficient Global Optimization) [11]. Kriging [12] is used as a surrogate and provides predictions as well as a measure of uncertainty for each point. The prediction and uncertainty

TABLE I
Categorization regarding the Surrogate's Role in an Optimization Algorithm.

Category	Algorithm / Study
Aided	MAES [19], SVC-DE [20]
Customized	MOEAD-EGO [26], K-RVEA [27], HSMEA [28], CSEA [29], PAL-SAPSO [30], CAL-SAPSO [31]
Centered	EGO [11], ParEGO [14], SMS-EGO [32], Max-Min SAEA [33], SACOBRA [34], SABLA [35], GS-MOMA [24], GSGA [25], MISO [36], GOSAC [37]
Just Once	[38], [39], [40]

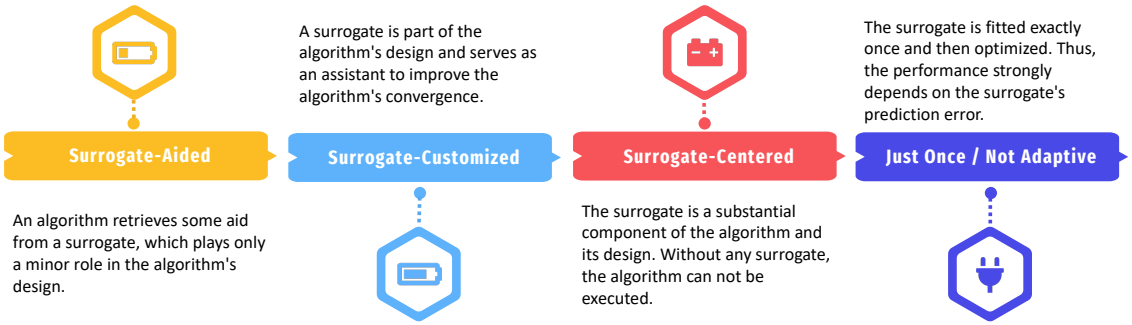

Fig. 2. Different Roles of Surrogates in the Design of an Algorithm.

together define the so-called acquisition function (or infill criterion), such as the expected improvement [11] or probability of improvement [41] aiming to balance exploitation and exploration simultaneously. The optimization of the acquisition function results in an infill solution, which is first evaluated and then added to the model. The procedure is repeated until a termination criterion is met. The limitation of finding only a single new solution in each iteration has been investigated thoroughly, and multi-point EGO approaches have been proposed [42], [43], [44]. Moreover, the concept has been generalized to solve multi-objective optimization problems by using decomposition [14], [26] or replacing the objective with a performance indicator based metric [32]. The idea has also been extended to handle constraints, which is especially important for solving real-world optimization problems [45]. Instead of using acquisition functions to address the surrogate's uncertainty, algorithms based on trust regions have been proposed. Inevitable, updating the trust-region radii becomes vital for the algorithm's performance [46]. Whereas original studies were limited to unconstrained single-objective optimization, the surrogate-assisted trust-region concept has been generalized to constrained and also multi-objective optimization [24], [25]. Apart from the approaches discussed above, the direct usage of surrogates in an algorithm has been explored in various areas, for instance, bi-level optimization [33], [35] or mixed-integer optimization [36]. All these approaches have in common that the algorithm has been designed with a strong dependency on the surrogate model. Thus, the surrogate's suitability and accuracy are critical for the optimization's success. Inaccurate surrogate predictions and error estimations, inevitably occurring in large-scale optimization problems, are known to be problematic [23].

In contrast to algorithms being designed based on surrogates, researchers have investigated surrogates' incorporation into existing optimization methods. Such approaches are also known as *surrogate-assisted* algorithms, emphasizing the surrogate's role as an assistant during optimization. In our categorization, surrogate-assisted algorithms are split up into two categories. On the one hand, algorithms can be aided by a surrogate where only minor changes of the original algorithm design are made; on the other hand, surrogate-customized methods where the algorithm has a significant impact on the algorithm's design. Because the judgment of *impact* is subjective, the transition between both categories is somewhat fluent.

The benefit of surrogate-aided algorithms is that with relatively minor modifications, a surrogate has been incorporated, and the performance has been improved [10]. One well-known approach is a pre-selection (or pre-filtering) which uses a surrogate to select a subset of solutions that usually would be evaluated on the EOP [19]. Moreover, instead of changing the behavior in a generation, surrogates have also been used across generations by switching between the expensive evaluation and surrogate predictions entirely for some iterations [15], [47]. Another example for a surrogate-influenced algorithm is modifying a memetic algorithm (genetic algorithm with local search) by executing the usually evaluation-intensive local search on the surrogate [48].

Besides surrogate-assisted methods with relatively minor modifications of existing algorithms, optimization methods can be customized to incorporate surrogate usage. This let arise surrogate-assisted variants of well-known algorithms, such as lqCMAES [49] derived from CMAES [50], KRVEA [27] and HSMEA [28] based on RVEA [51], MOEAD-EGO [26] as an improvement of MOEAD [52], or CAL-PSO [31] based on PSO [18] to name a few. Each surrogate-assisted variant is in principle based on an algorithm originally developed for computationally more inexpensive optimization problems but customizes the default behavior, for instance, by one or multiple local or global surrogates, implementing a sophisticated selection after surrogate-based optimization.

The increasing number of surrogate-assisted algorithms shows its importance and relevance in practice. Indisputably, approaches and ideas directly designed for and centered on one or multiple surrogates have their legitimacy but are somewhat tricky to use for newly proposed algorithms. The existence of many different surrogate-based algorithms also indicates the absence of a best practice procedure and the need for more generic methods. Thus, this study shall address this research gap and propose a surrogate-assisted framework of algorithms applicable to a broad category of optimization methods.

III. Methodology

One of the major challenges when proposing a generalized optimization framework is the number and strictness of assumptions being made. On the one hand, too many assumptions restrict the applicability; on the other hand, too few assumptions limit the usage of existing elements in algorithms.

Algorithm 1: Infill-And-Advance Interface

Input : Algorithm Φ

```
1 while Φ has not terminated do
2 |   X ← Φ.infill()
3 |   F, G ← evaluate(X)
4 |   Φ.advance(X, F, G)
5 end
```

In this study, we target any type of population-based algorithm with two phases in an iteration: the process of generating new solutions to be evaluated (infill) and a method processing evaluated infill solutions (advance). With these two methods, running an algorithm can be summarized by the pseudo-code shown in Algorithm 1. Until the algorithm Φ has been terminated, the *infill* method returns a set of new designs X to be evaluated. After obtaining the objective F and constraint G values for design, the algorithm is advanced by providing the evaluated solutions $\{X, F, G\}$. By looking at this interface, we further make two (weak) assumptions. First, we do not assume that X needs to be identical with the suggested designs from *infill* (Line 2 and 4), but can also be modified. Second, the *infill* method is non-deterministic, resulting in different designs X whenever called. Both assumptions can be considered weak because most population-based algorithms already fulfill them. So, how can existing optimization methods be described into *infill* and *advance* phases? Genetic algorithms (GAs) generate new solutions using evolutionary recombination-mutation operators and then process them using an environmental survival selection [16] operator; PSO methods create new solutions based on a particles' current velocity, personal best, and global best, and process the solutions using a replacement strategy [18]; CMAES samples new solutions from a normal distribution, which is then updated in each iteration [50]. Shown by well-known state-of-the-art algorithms following or being suitable to be implemented in this optimization method design pattern, this seems to be a reasonable assumption to be made for a generic framework. Moreover, it is worth noting that some researchers and practitioners also refer to the pattern as *ask-and-tell* interface.

However, how shall this interface now be utilized, and what role can surrogates play in improving the algorithm's performance? Precisely this is the subject of this article. Nevertheless, before moving on to the proposed framework, some more specifications of the surrogate usage are to be defined: First, the surrogate shall only be used as an assistant (in contrast to other methods where everything is developed centered around the surrogate). Second, the proposed method should be adaptive, allowing to decrease and increase the impact of surrogate usage and, if desired, even falling back to the original pseudo-code shown in Algorithm 1. Third, the surrogate prediction error needs to be addressed to ensure both exploitation and exploration. Altogether, the design goals are formulated to make the optimization framework and surrogate incorporation flexible. The proposed Generalized Probabilistic Surrogate-Assisted Framework (GPSAF) meets these goals by introducing two different phases: First, the α-phase using the current state of algorithm Φ introducing some surrogate influence. This pre-filtering phase uses a replacement strategy based on surrogate predictions; second, the β-phase continues to run algorithm Φ for *multiple* consecutive iterations on the surrogate resulting in a (convergence) search pattern. From the search pattern, solutions are selected by applying a probabilistic knockout tournament. The tournament incorporates the distribution of historical prediction errors by comparing

Algorithm 2: GPSAF: **G**eneralized **P**robablistic **S**urrogate-**A**ssisted **F**ramework

Input : Algorithm Φ, Surrogate Tournament Pressure α (≥ 1), Number of Simulated Iterations β (≥ 0), Replacement Probability Exponent γ, Maximum Number of Solution Evaluations $\texttt{SE}^{(\max)}$

```
/* Sample Design of Experiments (DOE)                    */
1  A ← ∅; P ← ∅; Q ← ∅; U ← ∅; e ← ∅
2  A.X ← doe();   A.F, A.G ← evaluate(A.X)
3  while size(A) < SE^(max) do
       /* Infill sols. from baseline algorithm          */
4      P.X ← Φ.infill()
       /* Estimate error - only initially               */
5      if e = ∅ then e ← estm_error(A.X, A.F, A.G);
       /* Surrogates for each obj. and constr.          */
6      S ← fit(A.X, A.F, A.G)
7      P.F̂, P.Ĝ ← S.predict(P.X)
       /* Surrogate Influence (α)                       */
8      foreach k ← 2 to α do
9          Q.X ← Φ.infill()
10         Q.F̂, Q.Ĝ ← S.predict(Q.X)
11         foreach j ← 1 to size(Q) do
12             if not dominates(P[j], Q[j]) then P[j] = Q[j] ;
13         end
14     end
       /* Surrogate Bias (β)                            */
15     Φ′ ← copy(Φ)
16     U ← ∅
17     foreach k ← 1 to β do
18         Q.k ← k
19         Q.X ← Φ′.infill()
20         Q.F̂, Q.Ĝ ← S.predict(Q.X)
21         foreach j ← 1 to size(Q) do
22             i ← closest(P.X, Q[j].X)
23             U[i] ← U[i] ∪ Q[j]
24         end
25         Φ′.advance(Q.X, Q.F̂, Q.Ĝ)
26     end
27     V ← list()
28     foreach j ← 1 to size(U) do
29         V ← V ∪ prob_knockout_tourn(U[j])
30     end
       /* Replacement (γ)                               */
31     foreach j ← 1 to size(P) do
32         ρ ← repl_prob(U[j], U, γ)
33         if rand() < ρ then P[j] ← V[j]) ;
34     end
       /* Evaluate on ESE                               */
35     P.F, P.G ← evaluate(P.X)
       /* Prepare next iteration of GPSAF               */
36     Φ.advance(P.X, P.F, P.G)
37     e ← update_error(P.F, P.G, P.F̂, P.Ĝ)
38     A ← A ∪ P
39 end
```

solutions under noise to address the surrogate inaccuracies. The two phases and their control parameter allow configuring the surrogate usage, and the probabilistic knockout tournament serves as a self-adaptive mechanism to balance its exploitation and exploration.

A. Generalized Probabilistic Surrogate-Assisted Framework

Before describing the responsibilities and details of each of the phases, the outline of the algorithm shall be discussed (see Algorithm 2). Before any surrogate can be fit, a solution archive A is initialized by some design of experiments `A.X` are generated in a space-filling manner. A good spread of solutions is recommended to allow surrogates to capture the overall fitness landscape as accurately as possible. `A.X` is evaluated on the expensive solution evaluation (ESE) resulting in `A.F` and `A.G` (Line 2). Then, while the number of evaluations is less than the maximum solution evaluation budget $\texttt{SE}^{(\max)}$, infill solutions `P.X` are generated by calling the non-deterministic *infill* method of the baseline algorithm Φ. The default execution of algorithm Φ would immediately evaluate `P.X` using ESE and directly feed the solutions back to the algorithm by executing $\Phi.\texttt{advance}(\texttt{P.X}, \texttt{P.F}, \texttt{P.G})$ (Line 35 and 36). However, instead of doing so, GPSAF modifies $P.X$ in a way to be influenced and biased by surrogates (Line 6 to 30) and advances the algorithm in the end of the iteration (Line 35 to 36). After having estimated the surrogate error and fitted the surrogates for objective and constraint functions, the α-phase adds surrogate influence to `P.X` by replacing solutions being predicted to be better (Line 8 to 14). Thereafter, the β-phase runs algorithm Φ for multiple generations (evaluations only on ASE) and assigns each solution to its closest `P.X`. For each of the resulting candidate solution pools `U[j]` assigned `P[j]` a probabilistic tournament determines the winning candidate (Line 15 to 30). Afterward, the replacement phase takes place where either the solution originating from the α-phase `P[j]` is kept or replaced with `U[j]` from the β-phase (Line 31 to 34). The solutions set to `P.X` are evaluated, and the algorithm Φ is advanced (Line 35 and 36). Finally, the prediction error is updated before starting the next iteration, and the newly evaluated solutions are added to archive A.

The overall outline of GPSAF shall provide an idea of where and when the α and β phases take place and what role they play in modifying the infill solutions fed back to the algorithm. Next, each phase shall be explained and discussed in detail.

B. Surrogate Influence through Tournament Pressure (α)

The first mechanism of GPSAF incorporates tournament pressure by utilizing the predictions of a surrogate as a referee. Tournament pressure is a well-known concept in evolutionary computation to introduce a bias towards more promising solutions [53]. Usually, its purpose is to introduce selection bias during mating to increase the chances of involvement of better-performing individuals. A specifically helpful control parameter is the number of competitors in each tournament to naturally increase or decrease the selection pressure.

Here, we borrow the tournament pressure mechanism to provide solutions with *surrogate* influence before their evaluation on the EOP. The surrogate predictions provide the necessary ASEs to determine the winner of each tournament. The number of competitors α in each tournament can control the amount of surrogate influence. In Figure 3 the surrogate-assisted tournament selection with three competitors ($\alpha = 3$) for four infill solutions ($n = 4$) is visualized. Initially, the algorithm's *infill* function is called three times to generate the solution sets X^{α_1}, X^{α_2}, and X^{α_3}. After evaluating each of the solution sets on the surrogate, a tournament takes place where α solutions of the j-th infill solution set X^{α_j} compete with each other. For instance, for the first tournament, the winner of $X_1^{\alpha_1}$, $X_2^{\alpha_1}$, and $X_3^{\alpha_1}$ is declared. The winner of each solution pool is determined as follows: if *all* solutions are infeasible, select the least infeasible solution; otherwise, select a non-dominated solution (break ties randomly). For both the constraint and objective values, only ASEs are used. By repeating the tournament n times and declaring the winners X_i where $i \in (1, \ldots, n)$, here X_1, X_2, X_3, and X_4, four surrogate-influenced solutions have been selected and the α-phase is completed.

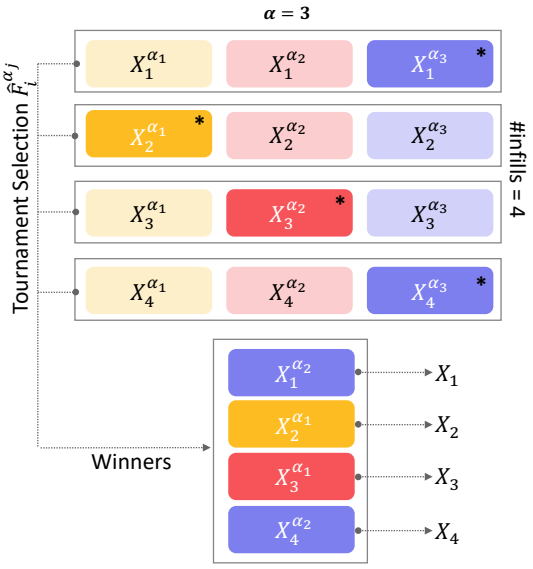

Fig. 3. Tournament selection with α competitors to create a surrogate-influenced infill solutions.

It is worth noting that because the *infill* method calls of algorithm Φ is non-deterministic and do not have any order, this can be implemented memory-friendly by a for loop as shown in Algorithm 2 (see Line 8 to 14). Generally, it shall also become apparent that setting $\alpha = 1$ disables the tournament selection and serves as a fallback to the original algorithm. By involving the surrogate in the tournament selection ($\alpha > 1$), the infill solutions `P` get a smaller or larger influence based on the number of competitors, which provides a natural inclusion of surrogate guidance.

C. Continue Optimization on Surrogate (β)

After completing the α-phase, the solution set `P` is already influenced by surrogates. But what are the limitations of the α-phase, and why is there a necessity for a second one? Even though the *infill* method is called multiple times, the algorithm

Algorithm 3: Probabilistic Knockout Tournament (PKT)

Input : Solution Set C, Prediction errors e, Number of winners k

1 $C^{(1)} \leftarrow \texttt{shuffle(C)}$
2 $t \leftarrow 1$
3 **while** $|C^{(t)}| > k$ **do**
4 **if** $|C^{(t)}|$ *is* odd **then** $C^{(t)} \leftarrow C^{(t)} \cup \texttt{rselect}(\texttt{C}^{(\texttt{t})}, 1)$;
5 $C^{(t+1)} \leftarrow \emptyset$
6 **foreach** $i \leftarrow 1$ **to** $|C^{(t)}|/2$ **do**
7 $w \leftarrow \texttt{compare\_noisy}(\texttt{C}^{(\texttt{t})}_{\texttt{2i}}, \texttt{C}^{(\texttt{t})}_{\texttt{2i+1}}, \texttt{e})$
8 $C^{(t+1)} \leftarrow C^{(t+1)} \cup w$
9 **end**
10 $t \leftarrow t + 1$
11 **end**
12 **if** $|C^{(t)}| < k$ **then**
13 $C^{(t)} \leftarrow C^{(t)} \cup \texttt{rselect}(\texttt{C}^{(\texttt{t-1})} \setminus \texttt{C}^{(\texttt{t})}, |\texttt{C}^{(\texttt{t})}| - \texttt{k})$
14 **end**
15 **return** $C^{(t)}$

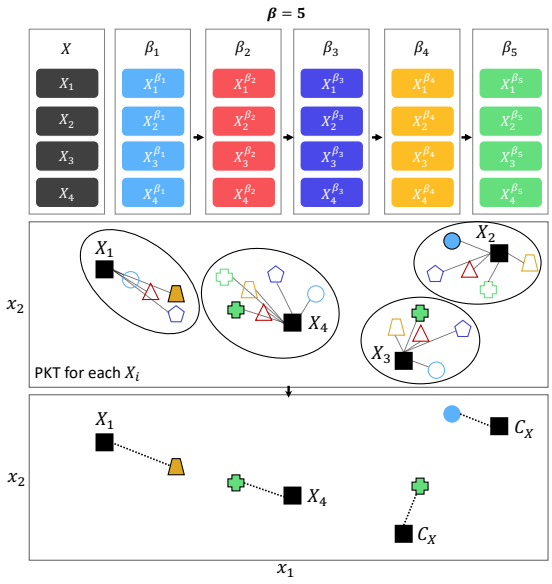

Fig. 4. Continue Running the Algorithm for β Iteration on the Surrogate.

is never *advanced* to the next iteration. Thus, it provides a *one step* look-ahead, which is not sufficient to find near-optimal solutions on the surrogate (which to some degree can be very useful for convergence). So, to further increase the surrogate's impact, the second phase looks β iterations into the future by calling infill *and* advance of the baseline algorithm repetitively. Whereas for a smaller β, the surrogates will be somewhat exploited, for a larger β, near-optimal solutions using ASEs will be found (similarly to optimizing the surrogate directly). However, it must be considered that ASEs have an underlying prediction error and can not be taken for granted. In GPSAF, the error is addressed by making not only use of the final solution set resulting from the β optimization runs but using the whole *search pattern*. The pattern is first divided into multiple clusters by assigning all solutions to their closest solution (in the design space) to each solution in P.X. Then, we use a so-called probabilistic knockout tournament (PKT) to select solutions from each cluster with the goal of self-adaptively exploiting surrogates. The goal is to use surrogates more when they provide accurate predictions but use them more carefully when they provide only rough estimations. Necessary for generalization, PKT also applies to problems with multiple objectives and constraints, often with varying complexities and surrogate errors to be considered.

Generally, we define PKT as a subset selection of k solutions from a set of solutions C by applying *pairwise* comparisons under noise as shown in Algorithm 3. Initially, the solution set C to select from is shuffled to randomize the matches (Line 1). If the current number of participants $|C^{(t)}|$ is odd, a random solution is chosen to compete twice (Line 4). Each competition occurs under noise, based on the current prediction error of the surrogates. The noise is added to each objective and constraint independently before comparing the solutions. After adding the noise, the comparison is identical to the subset selection explained in Section III-B (feasibility, dominance, random tie break) with two competitors ($\alpha = 2$). The winner of each round moves on to the next and is added to $C^{(t+1)}$ (Line 8). Finally, if too many solutions have been eliminated, randomly choose some losers from the last round (Line 13). This results in a set of solutions of size k being returned as tournament winners under noise. The design of PKT applies to the most general case of constrained multi-objective optimization because the selection procedure can be reduced two a comparison of two solutions.

Back to the cluster-wise selection in the β-phase where PKT is executed with $k = 1$ to obtain a winner for each solution set $\texttt{U}_\texttt{j}$. An example with five iterations ($\beta = 5$) and four infill solutions X_1, X_2, X_3, and X_4 is illustrated in Figure 4. Calling the *infill* and *advance* function of the baseline algorithm results in five solution sets (β_1 to β_5) with four solutions each. The advancement of multiple iterations is based on ASEs. In each iteration, all solutions are directly assigned to the closest X_i solution from the α-phase forming the cluster U_i. The cluster search pattern division is essential to preserve diversity. For each cluster, a winner V_i is declared by performing the PKT. For instance, in this example, X_1 has four solutions in U_1 where one from the fourth iteration β_4 is finally selected. At the end of the β-phase, each cluster U_i has at most one solution V_i to be assigned to (some clusters may stay empty because no solutions are assigned to it).

The β-phase exploits the surrogates significantly more than the α-phase by optimizing multiple iterations on the surrogates. In addition, mechanisms such as cluster-based search pattern selection help preserve diversity. Finally, it is worth noting that, analogously to the α-phase, the surrogate assistance can be disabled by setting a specific configuration ($\beta = 0$). Thus, GPSAF provides a fallback mechanism to the baseline optimization method without surrogate assistance (when $\alpha = 1$ and $\beta = 0$). Increasing one or the other will add more and more guidance through surrogate models. How the two phases, each resulting in a set of solutions, are now combined to find a trade-off between the more explorative first and more exploiting second phase shall be discussed next.

D. Balancing the Exploration and Exploitation (γ)

The α and β phases are designed to add surrogate assistance to an algorithm. The β phase utilizes the search pattern on the surrogate and assigns solutions from the α-phase. This means one can now choose to either stick to the more explorative α or exploit the β solution. The most simple way of making this choice is by replacing the solution with probability ρ. However, in a pilot study addressing single-objective optimization, it has been shown that a more dynamic selection strategy is beneficial [54].

A piece of particularly useful information for making this decision is the distribution of assigned solutions across clusters. The search pattern derived from surrogates with a high-density area indicates a region of interest. Thus, we propose to set the replacement probability:

$$\rho = \left(\frac{|U_j|}{\max_j |U_j|} \right)^{\gamma} \tag{1}$$

The denominator $\max_j |U_j|$ normalizes the number of assigned points with respect to the points in the current cluster $|U_j|$. The exponent γ can be used to control the importance of the distribution and was kept constant at $\gamma = 0.5$. The cluster with the highest density is always chosen from the β-phase because the nominator and denominator will be equal. This will necessarily be the case for baseline algorithms returning only one infill solution where a stronger surrogate bias is desirable. After the replacement, the solutions will finally be sent to the time-consuming solution evaluation.

E. Surrogate Management

Besides using surrogates in an algorithmic framework, some more words need to be said about the models themselves. First, one shall note that only the predictions of data points need to be provided by surrogates and no additional error estimation (the error estimates are kept track of by our method directly). Not requiring an error estimation does not limit the models to a specific type, unlike other surrogate-based algorithms. Second, each of the objective and constraint functions is modeled independently, known as *M1* in the surrogate usage taxonomy in [55]. Even though modeling all functions increases the algorithmic overhead, it prevents larger prediction errors through complexity aggregations of multiple functions. Third, a generic framework for optimizing computationally expensive functions requires a generic surrogate model implementation. Clearly, some model types are more suitable for some problems than others. Thus, to provide a more robust framework, each function is approximated with a set of surrogates, and the best one is finally chosen to be used. The surrogate types in this paper consist of the model types RBF [56] and Kriging [12], both initialized with different hyper-parameters (normalization, regressions, kernel). A pre-normalization step referred to as PLOG [34] is attempted and selected if well-performing for constraint functions. Two metrics assess the performance of a model: First, Kendall Tau Distance [57] comparing the ranking of solutions being less sensitive to outliers with a large prediction error; second, the Maximum Absolute Error (MAE) to break any ties. The value of MAE is also used as an error approximation when noise is added to individuals. The error estimation in the first iteration is based on k-fold cross-validation ($k = 5$) to get a rough estimate of how well a surrogate can capture the function type. The performance metrics are updated in each iteration by taking all solutions seen so far as training and the newly evaluated solutions as a test set. Finally, a moving average of five iterations to avoid a smooth and more robust estimation provides the data for selecting the best surrogate and estimating the prediction error for each objective and constraint.

IV. Experimental Results

In this section, we present the performance of GPSAF applied to various population-based algorithms solving unconstrained and constrained, single- and multi-objective optimization problems. Proposing an optimization framework requires comparing a group of algorithms, which is not a trivial task itself. Benchmarking is further complicated when non-deterministic algorithms are compared, in which case not only a single but multiple runs need to be considered.

For a fair comparison of optimization methods across test problems and to measure the impact of GPSAF on a baseline algorithm, we use the following ranking-based procedure:

i. Statistical Domination: After collecting the data for each test problem and algorithm ($\mathcal{A} \in \Omega$) from multiple runs, we perform a pairwise comparison of performance indicators (PI) between all algorithms using the Wilcoxon Rank Sum Test ($\alpha = 0.05$). The null-hypothesis H_0 is that no significant difference exists, whereas the alternative hypothesis is that the performance indicator of the first algorithm ($\texttt{PI}(\mathcal{A})$) is smaller than the one of the second one ($\texttt{PI}(\mathcal{B})$) The PI is for single-objective optimization the gap to the optimum (if known) or the best function value found. For multi-objective optimization IGD [58] (if optimum is known) or Hypervolume [59] is used.

$$\phi(\mathcal{A}, \mathcal{B}) = \textbf{RANKSUM}(\texttt{PI}(\mathcal{B}), \texttt{PI}(\mathcal{A}), \texttt{alt} =' \texttt{less}'), \tag{2}$$

where the function $\phi(\mathcal{A}, \mathcal{B})$ returns zero if the null hypothesis is accepted or a one if it is rejected.

ii. Number of Dominations: The performance $P(\mathcal{A})$ of algorithm $\mathcal{A}$ is then determined by the number of methods that are dominating it:

$$P(\mathcal{A}) = \sum_{\substack{\mathcal{B} \in \Omega \\ \mathcal{A} \neq \mathcal{B}}} \phi(\mathcal{B}, \mathcal{A}) \tag{3}$$

This results in a domination number $P(\mathcal{A})$ for each method, which is zero if no other algorithm does not outperform it.

iii. Ranking: Finally, we sort the methods by their $P(\mathcal{A})$. This may result in a partial ordering with multiple algorithms with the same $P(\mathcal{A})$ values. In order to keep the overall sum of ranks equal, we assign their average ranks in case of ties. For instance, let us assume five optimizations methods A, B, C, D, and E: algorithm A outperforms all others; between the performances of B,

TABLE II
A COMPARISON OF DE, GA, PSO, AND CMAES WITH THEIR GPSAF VARIANTS ON UNCONSTRAINED SINGLE-OBJECTIVE PROBLEMS WITH FOUR OTHER SURROGATE-ASSISTED ALGORITHMS. THE RANK OF THE BEST PERFORMING ALGORITHM IN EACH GROUP IS SHOWN IN BOLD. THE OVERALL BEST PERFORMING ALGORITHM FOR EACH PROBLEM IS HIGHLIGHTED WITH A GRAY SHADE.

Problem	DE	GPSAF-DE	GA	GPSAF-GA	PSO	GPSAF-PSO	CMAES	GPSAF-CMAES	SACOSO	SACC-EAM-II	SADESammon	SAMSO
f01	11.0	**4.0**	7.5	**2.5**	5.5	**1.0**	5.5	**2.5**	**7.5**	9.5	12.0	9.5
f02	10.0	**4.0**	4.0	**1.0**	4.0	**2.0**	7.5	**6.0**	12.0	**7.5**	10.0	10.0
f03	8.5	**3.0**	5.0	**1.5**	7.0	**1.5**	8.5	**5.0**	10.0	**5.0**	11.5	11.5
f04	8.5	**4.0**	4.0	**2.0**	4.0	**1.0**	**6.5**	**6.5**	10.0	**8.5**	11.0	12.0
f05	5.5	**1.0**	7.5	**3.0**	5.5	**2.0**	7.5	**4.0**	11.0	**9.5**	**9.5**	12.0
f06	**7.0**	**7.0**	**2.0**	**2.0**	**4.5**	**4.5**	9.0	**2.0**	10.0	**7.0**	11.0	12.0
f07	9.5	**5.5**	5.5	**2.0**	5.5	**2.0**	5.5	**2.0**	9.5	**8.0**	11.0	12.0
f08	8.0	**6.0**	8.0	**2.0**	5.0	**2.0**	4.0	**2.0**	10.0	**8.0**	11.0	12.0
f09	10.0	**3.5**	8.0	**3.5**	**3.5**	**3.5**	7.0	**3.5**	**3.5**	9.0	11.5	11.5
f10	10.5	**5.5**	5.5	**1.5**	5.5	**1.5**	**5.5**	**5.5**	10.5	**5.5**	10.5	10.5
f11	10.5	**3.5**	7.0	**1.5**	7.0	**3.5**	**7.0**	**7.0**	10.5	**1.5**	12.0	7.0
f12	**2.5**	6.5	**6.5**	**6.5**	**2.5**	**2.5**	**2.5**	6.5	10.0	**9.0**	11.0	12.0
f13	7.5	**5.0**	7.5	**2.5**	5.0	**1.0**	5.0	**2.5**	10.0	**9.0**	11.0	12.0
f14	9.5	**5.0**	7.5	**2.0**	5.0	**2.0**	5.0	**2.0**	9.5	**7.5**	11.0	12.0
f15	10.0	**3.5**	6.5	**1.5**	6.5	**1.5**	6.5	**3.5**	9.0	**6.5**	11.0	12.0
f16	9.0	**6.0**	6.0	**2.0**	9.0	**2.0**	4.0	**2.0**	11.5	**6.0**	11.5	9.0
f17	9.5	**5.0**	7.0	**3.0**	7.0	**3.0**	3.0	**1.0**	11.0	**7.0**	9.5	12.0
f18	9.5	**4.0**	6.0	**2.0**	6.0	**2.0**	6.0	**2.0**	9.5	**8.0**	11.0	12.0
f19	11.0	**5.0**	10.0	**1.0**	**5.0**	**5.0**	8.5	**5.0**	5.0	**2.0**	8.5	12.0
f20	9.5	**2.5**	7.0	**2.5**	**2.5**	**2.5**	**5.5**	**5.5**	9.5	**8.0**	11.5	11.5
f21	8.5	**7.0**	**3.5**	**3.5**	**3.5**	**3.5**	**3.5**	**3.5**	10.0	**8.5**	11.5	11.5
f22	9.5	**6.0**	6.0	**2.5**	6.0	**2.5**	**2.5**	**2.5**	9.5	**8.0**	11.0	12.0
f23	10.0	**5.5**	5.5	**1.5**	**5.5**	**5.5**	**11.0**	12.0	9.0	**1.5**	5.5	5.5
f24	10.0	**2.0**	8.0	**2.0**	4.0	**2.0**	8.0	**5.5**	8.0	**5.5**	11.0	12.0
Total	8.958	**4.583**	6.292	**2.292**	5.188	**2.479**	6.021	**4.146**	9.417	**6.896**	10.667	11.062

C, and D, no significant difference exists; E performs the worst. In this case, method A gets rank 1, the group of methods B, C, and D, rank $(2+3+4)/3 = 9/3 = 3$, and E rank 5. Averaging the ranks for ties penalizes an optimization method for being dominated by the same amount of algorithms as others and keeps the rank sum for each problem the same.

This conveniently provides a ranking for each test problem. To evaluate the performance of a method on a test suite, we finally average the ranks across problems. If an algorithm fails to solve a specific problem for all runs, it gets the maximum rank and becomes the worst performing algorithm. Otherwise, all failing runs will be ignored (this has only rarely happened for a competitor algorithm to compare with). The ranks shall be used to compare the performances of methods in this manuscript, the values of the performance indicators for the methods on all test problems can be found in the Supplementary Document. Each algorithm has been executed 11 times on each test problem. If not explicitly mentioned in the specific experiment, the total number of solution evaluations has been set to $\mathtt{SE}^{(\max)} = 300$. For some simpler constrained problems, even fewer evaluations have been used. A relatively limited evaluation budget also means that more complicated problems might not be solved (near) optimally. However, a comparison of how well an algorithm has performed shall imitate the situation researchers face in practice. If the number of variables is not fixed, the number of variables is fixed to 10. The results are presented in ranking tables where the overall best performing algorithm(s) are highlighted with a gray cell background for each ranking-based comparison for a test problem. The best-performing ones in a group are shown in bold.

Moreover, some more details about our implementation shall be said. For the baseline algorithms, we use implementations of population-based algorithms available in the well-known multi-objective optimization framework pymoo[1] [60] developed in Python. For all methods, the default parameters provided by the framework are kept unmodified, except the population size (=20) and the number of offsprings (=10) to create a more greedy implementation of the methods. The surrogate implementation of Kriging is based on a Python clone[2] of DACEFit [61] originally implemented in Matlab. The RBF models are a re-implementation based on [34]. The hyper-parameters of GPSAF were determined through numerous empirical experiments during the algorithm development. A reasonable and well-performing configuration given by $\alpha = 30$, $\beta = 5$, and $\gamma = 0.5$ is fixed throughout all experiments.

A. (Unconstrained) Single-objective Optimization

The first experiment investigates the capabilities of GPSAF for improving the performance of existing algorithms on unconstrained single-objective problems. We use the BBOB test problems (24 functions in total) available in the COCO-platform [62] which is a widely used test suite with a variety of more and less complex problems. Four well-known population-based optimization methods, DE [63], GA [16], PSO [18], and CMAES [50] serve as baseline optimization

[1] http://pymoo.org (Version 0.5.0)
[2] https://pypi.org/project/pydacefit/

TABLE III
A COMPARISON OF DE, GA, PSO, AND ISRES WITH THEIR GPSAF VARIANTS, ON CONSTRAINED SINGLE-OBJECTIVE PROBLEMS WITH SACOBRA – THE CURRENT STATE-OF-ART ALGORITHMS FOR CONSTRAINED OPTIMIZATION.

Problem	$\mathtt{SE}^{(\max)}$	DE	GPSAF-DE	GA	GPSAF-GA	PSO	GPSAF-PSO	GPSAF-ISRES	SACOBRA
G1	75	5.5	**3.5**	7.5	**3.5**	7.5	**5.5**	**1.0**	**2.0**
G2	300	**3.5**	7.0	**1.0**	3.5	6.0	**3.5**	**8.0**	**3.5**
G4	75	6.5	**3.5**	6.5	**5.0**	8.0	**3.5**	**1.5**	**1.5**
G6	75	7.0	**4.0**	7.0	**4.0**	7.0	**4.0**	**1.5**	**1.5**
G7	75	7.0	**4.0**	7.0	**4.0**	7.0	**4.0**	**2.0**	**1.0**
G8	100	7.0	**4.5**	7.0	**4.5**	7.0	**3.0**	**1.0**	**2.0**
G9	300	7.0	**4.5**	7.0	**2.5**	4.5	**2.5**	**7.0**	**1.0**
G10	300	8.0	**3.5**	6.5	**3.5**	6.5	**3.5**	**3.5**	**1.0**
G11	300	7.0	**2.5**	5.5	**2.5**	5.5	**2.5**	**2.5**	**8.0**
G12	300	6.0	**4.5**	8.0	**4.5**	7.0	**3.0**	**1.5**	**1.5**
G16	300	5.5	**2.5**	**5.5**	8.0	7.0	**2.5**	**2.5**	**2.5**
G18	300	7.0	**3.5**	7.0	**3.5**	7.0	**3.5**	**3.5**	**1.0**
G19	300	7.5	**2.0**	6.0	**2.0**	7.5	**2.0**	**5.0**	**4.0**
G24	300	7.5	**4.5**	7.5	**4.5**	6.0	**2.0**	**2.0**	**2.0**
Total		6.571	**3.857**	6.357	**3.964**	6.679	**3.214**	**3.036**	**2.321**

algorithms and their GPSAF variants provide a surrogate-assisted version. The results are compared with four other surrogate-assisted algorithms, SACOSO [22], SACC-EAM-II [64], SADESammon [65], SAMSO [66] available in the PlatEMO [67] framework. The rankings from the experiment are shown in Table II. First, one can note that GPSAF outperforms the other four existing surrogate-assisted algorithms. One possible reason for the significant difference could be their development for a different type of test suite (for instance, problems with a larger number of variables). In this test suite, some problems are rather complicated, and exploiting the surrogate too much will cause to be easily trapped in local optima. Also, we contribute the efficiency of GPSAF to the significant effort for finding the most suitable surrogate. The order of relative rank improvement is given by GA ($6.292/2.292 = 2.7452$), PSO (2.0927), DE (1.9546), and for CMAES (1.4522). Besides GPSAF-GA having the biggest relative rank improvement, it also is the overall best performing algorithm in this experiment, closely followed by GPSAF-CMAES. Altogether, a significant and quite remarkable improvement is achieved by applying GPSAF for (unconstrained) single-objective optimization.

B. Constrained Single-objective Optimization

Rarely are optimization problems unconstrained in practice. Thus, especially for surrogate-assisted methods aiming to solve computationally expensive real-world problems, the capability of dealing with constraints is essential. The so-called G-problems or G-function benchmark [68], [69] was proposed to develop optimization algorithms dealing with different kinds of constraints regarding the type (equality and inequality), amount, complexity, and result in feasible and infeasible search space. The original 13 test functions were extended in a CEC competition in 2006 [70] to 24 constrained single-objective test problems [71]. In this study, G problems with only inequality constraints (and no equality constraints) are used. Besides the GPSAF variants of DE and GA, improved stochastic ranking evolutionary strategy (ISRES) [72] is applied to GSPAF. ISRES implements an improved mating strategy using differentials between solutions in contrast to its predecessor SRES [73]. ISRES follows the well-known 1/7 rule, which means with a population size of μ individuals $7 \cdot \mu$ offsprings are created. For this study, GPSAF creates a steady-state variant of ISRES by using the proposed probabilistic knockout tournament to choose *one* out of the λ solutions. This ensures a fair comparison with SACOBRA [34] which also evaluates one solution per iteration. To the best of our knowledge, SACOBRA implemented in R [74] is currently the best-performing algorithm on the G problem suite.

The constrained single-objective results are presented in Table III. First, it is apparent that the GPSAF variants improve the baseline algorithms. Only for G2, the genetic algorithm outperforms its and other surrogate-assisted variants, which we contribute to the very restricted feasible search space (also, this has shown to be a difficult problem for surrogate-assisted algorithms in [34]). Second, GPSAF-ISRES shows the best results out of all GPSAF variants. This indicates that it is beneficial if the baseline method has been proposed with a specific problem class in mind. Even though DE, GA, and PSO can handle constraints (for instance, naively using the parameter-less approach), there are known to not perform particularly well on complex constrained functions without any modifications. In contrast, ISRES has been tested on the G problems in the original study and proven to be effective. Furthermore, adding surrogate assistance to it has further improved the results. Third, GPSAF-ISRES shows competitive performance to the state-of-the-art algorithm SACOBRA. In this experiment, out of all 14 test problems: GPSAF variants were able to outperform SACOBRA four times and a baseline algorithm (GA) one time; five times the performance of at least one GPSAF variant was similar; four times SACOBRA has shown significantly better results. Altogether, one can say GPSAF has created surrogate-assisted methods competing with the state-of-the-art method for constrained single-objective problems.

TABLE IV
A COMPARISON OF NSGA-II, SMS-EMOA, AND SPEA2 WITH THEIR GPSAF VARIANTS WITH FOUR SURROGATE-ASSISTED ALGORITHMS ON BI-OBJECTIVE OPTIMIZATION PROBLEMS.

Problem	NSGA-II	GPSAF-NSGA-II	SMS-EMOA	GPSAF-SMS-EMOA	SPEA2	GPSAF-SPEA2	AB-SAEA	K-RVEA	ParEGO	CSEA
ZDT1	9.0	**2.5**	9.0	**2.5**	9.0	**2.5**	6.0	5.0	**2.5**	7.0
ZDT2	9.0	**1.5**	9.0	**6.0**	9.0	**5.0**	3.0	4.0	**1.5**	7.0
ZDT3	9.0	**4.5**	9.0	**4.5**	9.0	**4.5**	4.5	**1.0**	2.0	7.0
ZDT4	6.5	**2.0**	6.5	**2.0**	6.5	**2.0**	9.0	6.5	10.0	**4.0**
ZDT6	7.5	**3.0**	9.5	**6.0**	9.5	**3.0**	5.0	3.0	**1.0**	7.5
WFG1	9.0	**6.0**	9.0	**6.0**	9.0	**6.0**	4.0	**2.0**	**2.0**	**2.0**
WFG2	7.0	**1.5**	**8.0**	9.0	**4.5**	**4.5**	4.5	**1.5**	10.0	4.5
WFG3	8.0	**3.5**	10.0	**3.5**	8.0	**1.5**	5.5	**1.5**	5.5	8.0
WFG4	6.5	**1.5**	9.0	**5.0**	6.5	**1.5**	**3.5**	**3.5**	9.0	9.0
WFG5	8.0	**2.5**	8.0	**4.0**	10.0	**5.0**	6.0	2.5	**1.0**	8.0
WFG6	9.0	**2.0**	10.0	**6.5**	4.5	**2.0**	**2.0**	4.5	6.5	8.0
WFG7	5.5	**4.0**	8.5	**2.0**	8.5	**2.0**	5.5	7.0	**2.0**	10.0
WFG8	7.5	**2.5**	10.0	**5.0**	9.0	**2.5**	**2.5**	6.0	**2.5**	7.5
WFG9	7.5	**3.0**	7.5	**3.0**	6.0	**3.0**	**3.0**	10.0	9.0	**3.0**
Total	7.786	**2.857**	8.786	**4.643**	7.786	**3.214**	4.571	**4.143**	4.607	6.607

TABLE V
A COMPARISON OF NSGA-III, SMS-EMOA, AND SPEA2 WITH THEIR GPSAF VARIANTS WITH FOUR SURROGATE-ASSISTED ALGORITHMS ON THREE-OBJECTIVE OPTIMIZATION PROBLEMS.

Problem	NSGA-III	GPSAF-NSGA-III	SMS-EMOA	GPSAF-SMS-EMOA	SPEA2	GPSAF-SPEA2	AB-SAEA	K-RVEA	ParEGO	CSEA
DTLZ1	**1.5**	3.5	**1.5**	6.5	**5.0**	8.0	10.0	9.0	6.5	**3.5**
DTLZ2	8.5	**2.0**	8.5	**2.0**	8.5	**2.0**	6.0	**4.0**	5.0	8.5
DTLZ3	**2.0**	5.5	**2.0**	**2.0**	**5.5**	8.0	10.0	9.0	**5.5**	**5.5**
DTLZ4	**6.5**	**6.5**	9.0	**6.5**	6.5	**3.5**	2.0	**1.0**	10.0	3.5
DTLZ5	6.0	**3.0**	9.0	**1.0**	9.0	**2.0**	6.0	6.0	**4.0**	9.0
DTLZ6	7.5	**5.5**	7.5	**3.5**	9.0	**5.5**	**1.5**	**1.5**	10.0	3.5
DTLZ7	8.0	**3.5**	8.0	**3.5**	8.0	**3.5**	3.5	**1.0**	10.0	6.0
Total	5.714	**4.214**	6.5	**3.571**	7.357	**4.643**	5.571	**4.5**	7.286	5.643

C. (Unconstrained) Multi-objective Optimization

Many applications have not one but multiple conflicting objectives to optimize. For this reason, this experiment focuses specifically on multi-objective optimization problems. As a test suite, we choose ZDT [75], a well-known test suite proposed when multi-objective optimization has gained popularity. Throughout this experiment, we set the number of variables to 10, except for the high multi-modal problem, ZDT4, where the number of variables is limited to 5. The WFG [76] test suit provides even more flexibility by being scalable with respect to the number of objectives. Here, we simply set the objective number to be two to create another bi-objective test suite. Moreover, the number of variables has been set to 10 where four of them are positional. The baseline algorithms NSGA-II [77], SMS-EMOA [78], and SPEA2 [79] are used as baseline algorithms. The results are compared with four other surrogate-assisted algorithms: AB-SAEA [80], KRVEA [27], ParEGO [14], CSEA [29] available in PlatEMO [67].

The results on the two multi-objective test suites are shown in Table IV. First, one can note that all surrogate-assisted algorithms outperform the ones without. This indicates that surrogate assistance effectively improves the convergence behavior. Second, GPSAF-NSGA-II performs the best with a rank of 2.893 and shows the best performance, followed by GPSAF-SPEA2, GPSAF-SMS-EMOA, and KRVEA. It is worth noting that ParEGO is penalized by being terminated for ZDT4 and WFG2, where the surrogate model was not able to be built.

To show the behavior of three-objective optimization problems, we have replaced NSGA-II with NSGA-III and run all algorithms on the DTLZ problems suite [81] test suite. The results are shown in Table V. Whereas for most problems, the GPSAF variants outperform the baseline algorithms, for DTLZ1 and DTLZ3, this is not the case. Both problems consist of multi-modal convergence functions, which causes a large amount of surrogate error. Thus, surrogate-assisted algorithms (including the four GPSAF is compared to) are misguided. This seems to be a vital observation deserving to be investigated in more detail into the future. Nevertheless, GPSAF improves the performance of baseline algorithms for the other problems. GPSAF-SMS-EMOA shows overall the best results in this experiment with an average rank of 2.786 followed by GPSAF-NSGA-III.

D. Constrained Multi-objective Optimization

Lastly, we shall compare GPSAF on constrained multi-objective optimization problems which often occur in real-world optimization. The challenge of dealing with multiple objectives and constraints in combination with computationally

TABLE VI
A COMPARISON OF NSGA-III, SMS-EMOA, AND SPEA2 WITH THEIR GPSAF VARIANTS WITH FOUR SURROGATE-ASSISTED ALGORITHMS ON CONSTRAINED MULTI-OBJECTIVE OPTIMIZATION PROBLEMS.

Problem	$\mathtt{SE}^{(\max)}$	NSGA-II	GPSAF-NSGA-II	SMS-EMOA	GPSAF-SMS-EMOA	SPEA2	GPSAF-SPEA2	HSMEA
C1-DTLZ1	300	**4.0**	**4.0**	**4.0**	**4.0**	**4.0**	**4.0**	**4.0**
C2-DTLZ2	300	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**	**4.5**	**1.0**
C3-DTLZ4	300	5.0	**1.5**	**5.0**	**5.0**	**5.0**	**5.0**	**1.5**
BNH	100	5.5	**2.5**	7.0	**4.0**	5.5	**2.5**	**1.0**
SRN	100	6.0	**3.0**	6.0	**3.0**	6.0	**3.0**	**1.0**
TNK	100	6.0	**2.0**	6.0	**2.0**	6.0	**2.0**	**4.0**
OSY	300	5.0	**1.5**	5.0	**3.0**	5.0	**1.5**	**7.0**
Total		5.143	**2.714**	5.357	**3.643**	5.143	**3.214**	**2.786**

expensive solution evaluations truly mimics the complexity of industrial optimization problems. We have compared our results with HSMEA [28] a recently proposed algorithm for constraint multi-objective optimization. With consultation of the authors, some minor modifications of the publicly available source code had to be made for dealing with computationally expensive constraints – as this is an assumption made in this study. The results on CDTLZ [82], BNH [83], SRN [84], TNK [85], and OSY [86] are shown in Table VI. Again, one can observe that the GPSAF variants consistently improve the performance of the baseline optimization methods. The only exception is C1-DTLZ1, where all methods could find no feasible solution, and thus, an equal rank is assigned. We contribute this to the complexity of the test problems given by the constraint violation and the multi-modality of the objective functions. For OSY, TNK, the GPSAF variants show a significantly better performance than HSMEA; for C3-DTLZ4, the performance is similar; and for C2-DTLZ2, BNH, and TNK, it performs better. Altogether, GPSAF-NSGA-II can obtain a better rank than HSMEA, but it shall be fair to say that for three out of the seven constrained multi-objective optimization problems, HSMEA is the winner. Nevertheless, GPSAF improved the performance of baseline algorithms and showed competitive results to another surrogate-assisted optimization method.

V. CONCLUDING REMARKS

This article has proposed a generalized probabilistic surrogate-assisted framework applicable to any type of population-based algorithm. GSPAF incorporates two different phases to provide surrogate assistance, one considering using the current state of the baseline algorithm and the other looking at multiple iterations into the future. In contrast to other existing surrogate-assisted algorithms, the surrogate search is not reduced to the final solutions on the surrogate, but the whole search pattern is utilized. Solutions are selected using a probabilistic tournament that considers surrogate prediction errors for objectives and constraints from the search pattern. GPSAF has been applied to multiple well-known population-based algorithms proposed for unconstrained and constrained single and multi-objective optimization. We have provided comprehensive results on test problem suites indicating that GPSAF competes and outperforms existing surrogate-assisted methods. The combination of GPSAF creating well-performing surrogate-assisted algorithms with its *simplicity* and *broad* applicability is very promising.

The encouraging results provide scope for further exploring generalized surrogate-assisted algorithms. One main challenge of a generalized approach is the recommendation of hyper-parameter configurations (α, β, ρ, or γ). The parameters have been set through empirical experiments; however, through the broad applicability, different mechanisms of baseline algorithms on very different optimization problems make it difficult to draw generally valid conclusions. A more systemic and possibly resource-intensive study shall provide an idea of how different hyper-parameter settings impact the performance of different algorithms. In addition, experiments investigating the sensitivity shall be especially of interest.

The focus of this study was to explore different types of problems with multiple objectives and constraints. Thus the number of variables was kept relatively small as this is often the case for computationally expensive problems. Thus, even though the search space dimensions do not directly impact the idea proposed in this article, it shall be part of a future study of how surrogate assistance performs for large-scale problems. Moreover, the number of solution evaluations per run has been set to 300, which allows using all solutions exhaustively for modeling without a large modeling overhead. However, more solution evaluations might be feasible for mediocre expensive optimization problems.

Nevertheless, this extensive explorative study on the use of surrogates in single and multi-objective optimization with and without constraints has indicated a viable new direction in congruence with existing emerging studies for a generic optimization methodology.

REFERENCES

[1] P. C. Roy, A. Guber, M. Abouali, A. P. Nejadhashemi, K. Deb, and A. J. M. Smucker, "Simulation Optimization of Water Usage and Crop Yield Using Precision Irrigation," in *Evolutionary Multi-Criterion Optimization*, K. Deb, E. Goodman, C. A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P. Reed, Eds. Cham: Springer International Publishing, 2019, pp. 695–706.

[2] H. Yin, H. Fang, G. Wen, Q. Wang, and Y. Xiao, "An adaptive RBF-based multi-objective optimization method for crashworthiness design of functionally graded multi-cell tube," *Structural and Multidisciplinary Optimization*, vol. 53, no. 1, pp. 129–144, 2016.

[3] S. Lucidi, M. Maurici, L. Paulon, F. Rinaldi, and M. Roma, "A simulation-based multiobjective optimization approach for health care service management," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1480–1491, 2016.

[4] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proceedings of the genetic and evolutionary computation conference*, ser. GECCO '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 419–427. [Online]. Available: https://doi.org/10.1145/3321707.3321729

[5] J. D. Anderson and J. Wendt, *Computational fluid dynamics*. Springer, 1995, vol. 206.

[6] B. Szabó and I. Babuška, *Finite element analysis*. John Wiley & Sons, 1991.

[7] W. Luo, R. Yi, B. Yang, and P. Xu, "Surrogate-assisted evolutionary framework for data-driven dynamic optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 2, pp. 137–150, 2019.

[8] H. Wang and Y. Jin, "A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems," *IEEE Transactions on Cybernetics*, vol. 50, no. 2, pp. 536–549, 2020.

[9] S. Olafsson and J. Kim, "Simulation optimization," in *Proceedings of the winter simulation conference*, vol. 1, 2002, pp. 79–84.

[10] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61 – 70, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210650211000198

[11] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *J. of Global Optimization*, 1998.

[12] D. G. Krige, "A statistical approach to some basic mine valuation problems on the Witwatersrand, by D.G. Krige, published in the Journal, December 1951 : introduction by the author," 1951.

[13] R. T. Haftka, D. Villanueva, and A. Chaudhuri, "Parallel surrogate-assisted global optimization with expensive functions –a survey," *Structural and Multidisciplinary Optimization*, vol. 54, no. 1, pp. 3–13, 2016. [Online]. Available: https://doi.org/10.1007/s00158-016-1432-3

[14] J. Knowles, "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006. [Online]. Available: https://doi.org/10.1109/TEVC.2005.851274

[15] Yaochu Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 5, pp. 481–494, 2002.

[16] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[17] R. Storn, "On the usage of differential evolution for function optimization," in *Proceedings of north american fuzzy information processing*, 1996, pp. 519–523.

[18] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - international conference on neural networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[19] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou, "Metamodel—Assisted evolution strategies," in *Parallel problem solving from nature — PPSN VII*, J. J. M. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañas, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 361–370, tex.ids= 2002-emmerich-metamodel-assisted.

[20] X. Lu, K. Tang, and X. Yao, "Classification-assisted Differential Evolution for computationally expensive problems," in *2011 IEEE congress of evolutionary computation (CEC)*, 2011, pp. 1986–1993.

[21] Y. Wang, D. Yin, S. Yang, and G. Sun, "Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1642–1656, 2019.

[22] C. Sun, Y. Jin, R. Cheng, J. Ding, and J. Zeng, "Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 644–660, 2017, tex.ids= 2017-sun-coop-swarm.

[23] J. Stork, M. Friese, M. Zaefferer, T. Bartz-Beielstein, A. Fischbach, B. Breiderhoff, B. Naujoks, and T. Tušar, "Open issues in surrogate-assisted optimization," in *High-performance simulation-based optimization*, T. Bartz-Beielstein, B. Filipič, P. Korošec, and E.-G. Talbi, Eds. Cham: Springer International Publishing, 2020, pp. 225–244.

[24] D. Lim, Y. Jin, Y. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 329–355, 2010.

[25] X. Cai, L. Gao, and X. Li, "Efficient generalized surrogate-assisted evolutionary algorithm for high-dimensional expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 365–379, 2020.

[26] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2010. [Online]. Available: https://doi.org/10.1109/TEVC.2009.2033671

[27] T. Chugh, Y. Jin, K. Miettinen, J. Hakanen, and K. Sindhya, *K-rvea: A kriging-assisted evolutionary algorithm for many-objective optimization*, Mar. 2016.

[28] A. Habib, H. K. Singh, T. Chugh, T. Ray, and K. Miettinen, "A multiple surrogate assisted decomposition-based evolutionary algorithm for expensive Multi/Many-Objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 1000–1014, 2019. [Online]. Available: https://doi.org/10.1109/TEVC.2019.2899030

[29] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, and Y. Jin, "A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 74–88, 2019.

[30] Z. Lv, L. Wang, Z. Han, J. Zhao, and W. Wang, "Surrogate-assisted particle swarm optimization algorithm with Pareto active learning for expensive multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 838–849, 2019, tex.ids= 2019-lv-pal-pso.

[31] H. Wang, Y. Jin, and J. Doherty, "Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2664–2677, 2017.

[32] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, "Multiobjective optimization on a limited budget of evaluations using model-assisted $\mathcal{S}$-Metric selection," in *Parallel problem solving from nature –PPSN x*, G. Rudolph, T. Jansen, N. Beume, S. Lucas, and C. Poloni, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 784–794.

[33] Yew-Soon Ong, P. B. Nair, and K. Y. Lum, "Max-min surrogate-assisted evolutionary algorithm for robust design," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 392–404, 2006.

[34] S. Bagheri, W. Konen, M. Emmerich, and T. Bäck, "Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets," *Applied Soft Computing*, vol. 61, pp. 377 – 393, 2017.

[35] M. M. Islam, H. K. Singh, and T. Ray, "A surrogate assisted approach for single-objective bilevel optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 681–696, 2017.

[36] J. Müller, "MISO: mixed-integer surrogate optimization framework," *Optimization and Engineering*, vol. 17, no. 1, pp. 177–203, Mar. 2016. [Online]. Available: https://doi.org/10.1007/s11081-015-9281-2

[37] J. Müller and J. D. Woodbury, "GOSAC: global optimization with surrogate approximation of constraints," *Journal of Global Optimization*, vol. 69, no. 1, pp. 117–136, Sep. 2017. [Online]. Available: https://doi.org/10.1007/s10898-017-0496-y

[38] G. Lei, T. Wang, Y. Guo, J. Zhu, and S. Wang, "System-level design optimization methods for electrical drive systems: Deterministic approach," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 6591–6602, 2014.

[39] S. Stipetic, W. Miebach, and D. Zarko, "Optimization in design of electric machines: Methodology and workflow," in *2015 intl aegean conference on electrical machines power electronics (ACEMP), 2015 intl conference on optimization of electrical electronic equipment (OPTIM) 2015 intl symposium on advanced electromechanical motion systems (ELECTROMOTION)*, 2015, pp. 441–448.

[40] M. Dorica and D. Giannacopoulos, "Response clustering for electromagnetic modeling and optimization," *IEEE Transactions on Magnetics*, vol. 42, no. 4, pp. 1127–1130, 2006.

[41] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001. [Online]. Available: https://doi.org/10.1023/A:1012771025575

[42] F. Viana and R. Haftka, "Surrogate-based optimization with parallel simulations using the probability of improvement," in *13th AIAA/ISSMO multidisciplinary analysis optimization conference*, 2010.

[43] P. Beaucaire, C. Beauthier, and C. Sainvitu, "Multi-point infill sampling strategies exploiting multiple surrogate models," in *GECCO '19: Proceedings of the genetic and evolutionary computation conference companion*. New York, NY, USA: ACM, 2019, pp. 1559–1567.

[44] N. Berveglieri, B. Derbel, A. Liefooghe, H. Aguirre, Q. Zhang, and K. Tanaka, "Designing parallelism in surrogate-assisted multiobjective optimization based on decomposition," in *Proceedings of the 2020 genetic and evolutionary computation conference*, ser. GECCO '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 462–470. [Online]. Available: https://doi.org/10.1145/3377930.3390202

[45] S. Bagheri, W. Konen, R. Allmendinger, J. Branke, K. Deb, J. Fieldsend, D. Quagliarella, and K. Sindhya, "Constraint handling in efficient global optimization," in *Proceedings of the genetic and evolutionary computation conference*, ser. GECCO '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 673–680. [Online]. Available: https://doi.org/10.1145/3071178.3071278

[46] Y. S. Ong, P. B. Nair, and A. J. Keane, "Evolutionary optimization of computationally expensive problems via surrogate modeling," *AIAA Journal*, vol. 41, no. 4, pp. 687–696, 2003. [Online]. Available: https://doi.org/10.2514/2.1999

[47] A. Ratle, *Accelerating the convergence of evolutionary algorithms by fitness landscape approximation*, ser. International Conference on Parallel Problem Solving from Nature. Springer, 1998.

[48] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 1, pp. 66–76, 2007.

[49] N. Hansen, "A global surrogate assisted CMA-ES," in *Proceedings of the genetic and evolutionary computation conference*, ser. GECCO '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 664–672. [Online]. Available: https://doi.org/10.1145/3321707.3321842

[50] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, Jun. 2001. [Online]. Available: http://dx.doi.org/10.1162/106365601750190398

[51] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[52] Q. Zhang and H. Li, "A multi-objective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation, Accepted*, vol. 2007, 2007.

[53] B. L. Miller, B. L. Miller, D. E. Goldberg, and D. E. Goldberg, "Genetic Algorithms, Tournament Selection, and the Effects of Noise," *Complex Systems*, vol. 9, pp. 193–212, 1995.

[54] J. Blank and K. Deb, "PSAF: A Probabilistic Surrogate-Assisted Framework for Single-Objective Optimization," in *GECCO '21: Proceedings of the genetic and evolutionary computation conference companion*. New York, NY, USA: ACM, 2021, place: New York, NY, USA. [Online]. Available: https://doi.org/10.1145/3449639.3459297

[55] K. Deb, R. Hussein, P. C. Roy, and G. Toscano-Pulido, "A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 104–116, 2019.

[56] R. L. Hardy, "Multiquadric equations of topography and other irregular surfaces," *Journal of Geophysical Research (1896-1977)*, vol. 76, no. 8, pp. 1905–1915, Mar. 1971. [Online]. Available: https://doi.org/10.1029/JB076i008p01905

[57] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938. [Online]. Available: http://www.jstor.org/stable/2332226

[58] C. A. Coello Coello and M. Reyes Sierra, "A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm," in *MICAI 2004: Advances in artificial intelligence*, R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, and H. Sossa, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 688–697.

[59] E. Zitzler, D. Brockhoff, and L. Thiele, "The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators via Weighted Integration," in *Proceedings of the 4th International Conference on Evolutionary Multi-criterion Optimization*, ser. EMO'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 862–876. [Online]. Available: http://dl.acm.org/citation.cfm?id=1762545.1762618

[60] J. Blank and K. Deb, "pymoo: Multi-objective Optimization in Python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[61] S. Lophaven, H. B. Nielsen, and J. Søndergaard, "DACE – a MATLAB kriging toolbox," 2002.

[62] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, 2020.

[63] R. Storn and K. Price, "Differential Evolution –A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, no. 4, pp. 341–359, Dec. 1997.

[64] J. Blanchard, C. Beauthier, and T. Carletti, "A surrogate-assisted cooperative co-evolutionary algorithm using recursive differential grouping as decomposition strategy," in *2019 IEEE congress on evolutionary computation (CEC)*, 2019, pp. 689–696.

[65] G. Chen, K. Zhang, X. Xue, L. Zhang, J. Yao, H. Sun, L. Fan, and Y. Yang, "Surrogate-assisted evolutionary algorithm with dimensionality reduction method for water flooding production optimization," *Journal of Petroleum Science and Engineering*, vol. 185, p. 106633, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092041051931054X

[66] F. Li, X. Cai, L. Gao, and W. Shen, "A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1390–1402, 2021.

[67] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.

[68] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, Mar. 1996. [Online]. Available: https://doi.org/10.1162/evco.1996.4.1.1

[69] C. A. Floudas and P. M. Pardalos, "A collection of test problems for constrained global optimization algorithms," in *Lecture notes in computer science*, 1990.

[70] *IEEE international conference on evolutionary computation, CEC 2006, part of WCCI 2006, vancouver, BC, canada, 16-21 july 2006*. IEEE, 2006. [Online]. Available: https://ieeexplore.ieee.org/xpl/conhome/11108/proceeding

[71] J. Liang, T. Runarsson, E. Mezura-Montes, M. Clerc, P. Suganthan, C. Coello, and K. Deb, "Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization," *Nangyang Technological University, Singapore, Tech. Rep*, vol. 41, Jan. 2006.

[72] T. Runarsson and X. Yao, "Search biases in constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 2, pp. 233–243, 2005.

[73] ——, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.

[74] R Core Team, "R: A language and environment for statistical computing," Vienna, Austria, manual, 2021, organization: R Foundation for Statistical Computing. [Online]. Available: https://www.R-project.org/

[75] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000. [Online]. Available: https://doi.org/10.1162/106365600568202

[76] S. Huband, L. Barone, L. While, and P. Hingston, "A Scalable Multi-objective Test Problem Toolkit," in *Evolutionary Multi-Criterion Optimization*, C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 280–295.

[77] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Trans. Evol. Comp*, vol. 6, no. 2, pp. 182–197, Apr. 2002. [Online]. Available: https://doi.org/10.1109/4235.996017

[78] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221706005443

[79] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," 2001.

[80] X. Wang, Y. Jin, S. Schmitt, and M. Olhofer, "An adaptive Bayesian approach to surrogate-assisted evolutionary multi-objective optimization," *Information Sciences*, vol. 519, pp. 317–331, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025520300591

[81] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary multiobjective optimization: Theoretical advances and applications*, A. Abraham, L. Jain, and R. Goldberg, Eds. London: Springer London, 2005, pp. 105–145.

[82] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 602–622, Aug. 2014.

[83] T. T. Binh and U. Korn, "MOBES: A multiobjective evolution strategy for constrained optimization problems," in *In Proceedings of the Third International Conference on Genetic Algorithms*, 1997, pp. 176–182.

[84] N. Srinivas and K. Deb, "Multi-Objective function optimization using non-dominated sorting genetic algorithms," *Evolutionary Computation Journal*, vol. 2, no. 3, pp. 221–248, 1994.
[85] M. Tanaka, "GA-based decision support system for multi-criteria optimization," in *Proceedings of the international conference on systems, man and cybernetics*, vol. 2, 1995, pp. 1556–1561.
[86] A. Osyczka and S. Kundu, "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm," *Structural optimization*, vol. 10, no. 2, pp. 94–99, Oct. 1995. [Online]. Available: https://doi.org/10.1007/BF01743536

Julian Blank is a PhD candidate in the Department of Computer Science and Engineering at Michigan State University. He received his B.Sc. in Business Information Systems from Otto von Guericke University, Germany, in 2010. He was a visiting scholar for six months at the Michigan State University, Michigan, USA, in 2015, and finished his M.Sc. in Computer Science at Otto von Guericke University, Germany, in 2016. He is the leading developer of pymoo, an open-source multi-objective optimization framework in Python. His research interests include evolutionary computation, multi-objective optimization, surrogate-assisted optimization, and machine learning.

Kalyanmoy Deb is Fellow, IEEE, University Distinguished Professor and Koenig Endowed Chair Professor with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, Michigan, USA. He received his Bachelor's degree in Mechanical Engineering from IIT Kharagpur in India, and his Master's and Ph.D. degrees from the University of Alabama, Tuscaloosa, USA, in 1989 and 1991. He is largely known for his seminal research in evolutionary multi-criterion optimization. He has published over 580 international journal and conference research papers to date. His current research interests include evolutionary optimization and its application in design, modeling, AI, and machine learning. He is one of the top-cited EC researchers with more than 190,000 Google Scholar citations.